



95-702 Distributed Systems

Lecture 1: Five Videos

Course administration and a brief introduction to the material.

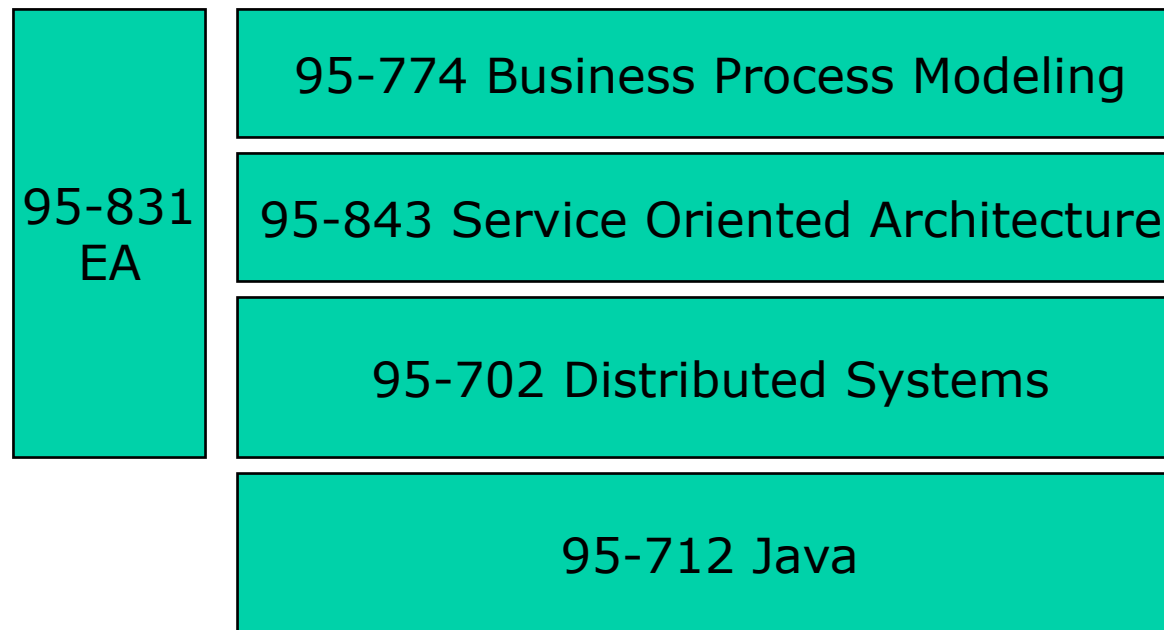
Instructors: Michael McCarthy & Joe Mertz

95-702 Distributed Systems
Master of Information System
Management

Course Web Site

- <http://www.andrew.cmu.edu/course/95-702/>
 - read the syllabus
 - read the course description
 - read the weekly schedule
 - * projects posted there
 - * course slides posted there
 - * project rubric is posted there
- Blackboard too!!
 - discussion board
 - grade postings

How is DS related to other courses?



What cool technologies will we use?

- IDE (Netbeans)
- Application Server (Glassfish)
- Web Services (REST and SOAP)
- Message Oriented Middleware (Sun's JMS Message Queue)
- Distributed Objects (Java RMI, and EJB's)
- Mobile platform (Android)
- Hadoop, MapReduce and Hive (Linux on Heinz cluster)

First Lab - This Week

- The first lab will cover instructions on getting started with the course technologies.
- The installation includes Netbeans, Glassfish and the Android emulator.
- You earn points for labs.
- This first lab will be for credit.
- Project 1 will be assigned soon.

Structure of the Course

- Lectures (many on video) with class time for activities
- Labs (with your active hands-on involvement)
- Projects (programming) **The secret is to start early.**
- Midterms (2)
- Final examination

Readings

- Readings from the required text are assigned for each lecture -- read them in advance.
- Readings from the web will also be assigned.
- For this week, read Coulouris chapters 1 and 2

Grading

- Programming Projects (6) 38%
- Midterm 1 10%
- Midterm 2 15%
- Final Exam 25%
- Labs (12) 12%
- Quizzes for practice 0%
- We will be very fussy about deadlines. One second late is late. See late assignment policy on syllabus.

Review of Syllabus

- Quizzes 0%
- Late projects, one week late free, one time, else -10% per day
- Projects are not treated the same as labs. Projects must be completed individually.
- Labs allow for team work and help from TA.
- Attendance at your scheduled lab is worth .25 points.
- Completion (within 6 days) of the lab is worth .75 points.
- Closed laptop policy
- Do not cheat, R grade is immediate. Don't share code with each other - we check! Don't copy code from or post code to external servers (e.g. GitHub) – we check.
- Use the discussion board
- Easy on the email
- If serious problem, contact us via email or phone
- TA's guide you but do not solve problems for you
- Exams take precedence over job interviews and travel
- Complaints about grading, see the rubric & the TA within one week
- See you in the next video...

How does program logic share data or communicate?


- | | Non-distributed | Distributed |
|---------------------------------------|-------------------|-----------------------|
| • Shared files | COBOL 2 COBOL | NFS, AFS, HDFS |
| • Shared database | DB2 Mainframe | OData |
| • Procedure Call or Method Invocation | Programming 101 | Java RMI, Web Service |
| • Messaging through a third party | Unix or DOS pipes | JMS or ESB's |

Fundamental characterization of distributed systems

- Components are located on networked computers and execute concurrently.
- Components communicate and coordinate only by passing messages
- Time differs on each system.
- Many challenges associated with distributed systems are not new...

The Pony Express

PONY EXPRESS!

CHANGE OF TIME!  REDUCED RATES!

10 Days to San Francisco!

LETTERS
WILL BE RECEIVED AT THE
OFFICE, 84 BROADWAY,
NEW YORK,
Up to 4 P. M. every TUESDAY.
AND
Up to 2½ P. M. every SATURDAY,
Which will be forwarded to connect with the PONY EXPRESS leaving
ST. JOSEPH, Missouri,
Every WEDNESDAY and SATURDAY at 11 P. M.

TELEGRAMS
Sent to Fort Kearney on the mornings of MONDAY and FRIDAY, will connect
with PONY leaving St. Joseph, WEDNESDAYS and SATURDAYS.

EXPRESS CHARGES.
LETTERS weighing half ounce or under..... \$1 00
For every additional half ounce or fraction of an ounce 1 00
In all cases to be enclosed in 10 cent Government Stamped Envelopes,
And all Express CHARGES Pre-paid.
PONY EXPRESS ENVELOPES For Sale at our Office.
WELLS, FARGO & CO., Ag'ts.
New York, July 1, 1861.
SLOPE & JAMES, SEEDINGERS AND PRINTERS, 9, FULTON STREET, NEW YORK

Issues:

- heterogeneity
- security
- reliability
- failure handling
- bandwidth
- latency

95-702 Distributed Systems

Master of Information System
Management

From Wikipedia

12

The Pony Express and The Telegraph

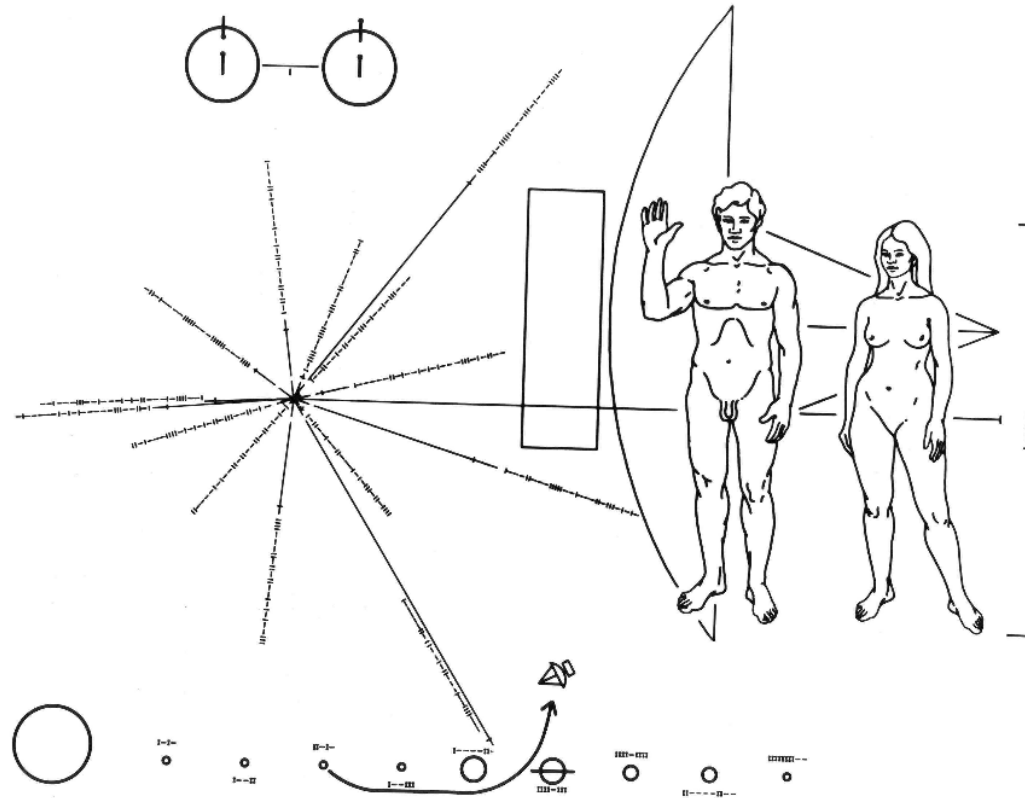


And one system may be replaced by another

Around 1860

Associated Press

Pioneer Plaque early 70's



An attempt at communicating a universal message.

Issues:
-reliability
-interoperability
-exemplifies a loosely coupled system
-would the arrow make sense?

What are some challenges in constructing DS?

- All the challenges associated with stand alone systems development **plus**
- **Heterogeneity** of components may hinder interoperability
- **Security** (Eve and Mallory on the wire)
- **Scalability** (we may need to ramp up by adding resources)
- **Failure handling** (networks, processors, remote systems)
- **Concurrency** of components adds complexity (e.g. several visits at once)
- **Openness** (Can we add to or modify the system)

What false assumptions may be made by designers?

- The network is secure.
- The network environment is homogeneous.
- Latency is zero.
- Bandwidth is infinite.
- Transport cost is zero.
- There is one administrator.
- The network is reliable.
- Components don't fail independently.

Why build these things?

- To communicate and share resources.
- We want to share:

Executable code – Javascript, MapReduce
Data (Odata)

CPU cycles (SETI search for aliens)

Documents (The original WWW)

Printers, Files (NFS, AFS)

Objects (Java RMI, EJB's, Enterprise Objects)

Services in a Service Oriented Architecture

Some Notes from Chapter Two

System Architecture From Chapter 2

Definition: The **architecture** of a system is its structure in terms of separately specified components and their interrelationships”.

Goal: The structure will meet present and future demands.

Concerns: reliability, manageability, adaptability, cost-effectiveness, security

Architectural Elements of a Distributed System

- Communicating entities
- Communication paradigms
- Roles played by communicating entities
- Placement of communication entities

Communicating Entities

- From a **system level**: Processes, threads or simply nodes are communicating.
- From a **problem level**: Objects, Components, Web Services are communicating.
- In **asynchronous systems**, the client makes a call and continues with other business. Perhaps it provides a means for a response.
- In **synchronous systems**, the client calls, blocks and waits for the response.

Architectural Elements of a Distributed System

- Communicating entities
- Communication paradigms
- Roles played by communicating entities
- Placement of communication entities

DS Communication Paradigms

Coupling is the degree to which some communicating entity makes assumptions about its partner.

Interprocess communication (TCP Sockets, UDP Sockets, Multicast Sockets)
Low level. Often use to build higher level abstractions. Coupled in time.

Remote invocation (Two way exchange with a remote operation, procedure or method) RPC, RMI, HTTP, DCOM, CORBA. Higher level abstractions.

Coupled in time (both parties exist during interaction)

Coupled in space (parties likely know who they are interacting with)

Indirect communication (less **tightly coupled** and involving a **third party**)
Communicating to a group by sending a message to a group identifier

Publish-subscribe (AKA distributed event based systems) routes messages to interested parties. One-to-many style of communication.

Message queues (AKA channels) for point-to-point messaging.

Tuple spaces allows for the placement and withdrawal of structured sequences of data.

Architectural Elements of a Distributed System

- Communicating entities
- Communication paradigms
- Roles played by communicating entities
- Placement of communication entities

Roles and Responsibilities

- Entities interact to perform a useful activity.
- One entity may act as a client and another as a server.
 - Request/Response
 - Request/Acknowledge
 - Request/Acknowledge/Poll
 - Request/Acknowledge/Callback
- Each entity may act as a peer.

Architectural Elements of a Distributed System

- Communicating entities
- Communication paradigms
- Roles played by communicating entities
- Placement of communication entities

Placement of Communicating Entities

- Entities may be placed on a single or multiple machines.
- Data may be cached and services replicated. Why replicate?
- Mobile code (e.g. applets, Java Script).
- Mobile agents or worms.

Architectural Patterns (1)

A **Layered architecture** is the vertical organization of services into layers of abstraction:

- * applications and services layered on the top.
- * middleware appears between the application and the operating system.
- * The operating system sits on top of the computer and network hardware.

Architectural Patterns (2)

A Tiered architecture:

- is complimentary to layering.
- is usually applied to the **applications and services** layer.
- is a technique to organize the functionality of a given layer and place this functionality into appropriate servers and onto physical devices.
- An application may be described in terms of **presentation logic, business logic, and data logic**.
- May partition an application into two tiers or three.
- Main driver: To promote **separation of concerns**.

Why is separation of concerns
so important?

95-702 Distributed Systems
Coulouris
5Ed.

Note: presentation logic may
present data to a non-human.

Architectural Patterns (3)

In a two-tier solution, the business logic and user interface may reside on the client and the data logic layer may be placed on the server. This is the classic client server architecture.

Other organizations are possible:

In a three-tier solution, the logical description may correspond directly to the physical machines and processes.

An AJAX application such as Google Maps is an example of a responsive multi-tiered application. New map tiles (256X256 pixel images) are fetched as needed.

The thin client approach is a trend in distributed computing. Move complexity into internet based services. Cloud computing and Virtual Network Computing (remote desktop) are examples.

Two commonly occurring architectural patterns in distributed systems

- The **proxy pattern**: the client makes calls on a local object (the proxy) that has the same interface as a remote object. The proxy hides the communication details.
- The **brokerage pattern** consists of a trio of service provider, service requestor and service broker (typically with lookup and bind operations).

Transparency

- Goal: To raise the level of abstraction by separation of concerns.

Transparency to raise the level of abstraction

- Types of Transparency (or concealment)

Access transparency: enables local and remote resources to be accessed using identical operations.

Location transparency: enables resources to be accessed without knowledge of their location.

Concurrency transparency: enables several processes to operate concurrently using shared resources without interference between them.

Replication transparency: enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.

Transparency to raises the level of abstraction

Failure transparency: enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.

Mobility transparency: allows the movement of resources and clients within a system without affecting the operation of users or programs.

Performance transparency: allows the system to be reconfigured to improve performance as loads vary.

Scaling transparency: allows the system and applications to expand in scale without change to the system structure or the application algorithms.